PARALLEL GLOBAL ELIMINATION ALGORITHM AND ARCHITECTURE DESIGN FOR FAST BLOCK MATCHING MOTION ESTIMATION

Yu-Wen Huang, Chen-Han Tsai, and Liang-Gee Chen

DSP/IC Design Lab., Graduate Institute of Electronics Engineering and
Department of Electrical Engineering, National Taiwan University
{yuwen, phenom, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

The critical path of the hardware for global elimination algorithm (GEA) is too long to meet the real-time constraints for high-end applications. In this paper, we propose a new parallel GEA and its corresponding architecture. By dividing candidate blocks into independent groups and finding the most probable candidates of each group in parallel, instead of sequentially searching within the whole search range, parallel design can be developed as an array of GEA processing elements with much shorter critical path. Besides, the GEA processing element is optimized to reduce 30% of the gates, and the 2-D data reuse is organized to save 80% of the SRAM bandwidth, which also reduces a lot of power. Simulation results show that our implementation can achieve real time processing of D1 30Hz video with search range as H[-64, +63.5] V[-32, +31.5] while the operating frequency is 70MHz, and the gate count is 113K. Compared with full search, our gate count is six times smaller under the same frequency, and the PSNR loss is at most 0.1-0.2dB.

1. INTRODUCTION

Motion estimation (ME) removes the temporal redundancy within frames and provides coding systems with high compression ratio. Block matching algorithm (BMA) is mostly selected as the ME module in video codecs because of its simplicity and good performance. Among all the block matching algorithms, full-search block matching algorithm (FSBMA) is the most popular but demands the most computation. For example, real-time ME for CIF (352×288) 30Hz video with [-16, +15] search range requires 9.3 Giga-operations per second (GOPS). If the frame size is enlarged to D1 (720×480) 30Hz with [-32, +31] search range, 127 GOPS is required. Such huge computational complexity is far beyond the processing capabilities of general purpose processors.

Successive elimination algorithm (SEA) [1][2] can reduce the heavy computation of FSBMA and maintain the same results as FSBMA. We proposed a global elimination algorithm (GEA) [3] to remove the branches and to make the data flow much more regular for hardware design. However, the drawback of our previous GEA architecture is the longer critical path. It is difficult to meet the real-time requirement for high specifications. In this paper, we propose a new parallel GEA and its corresponding architecture to solve the encountered problem. The rest of this paper is organized as follows. The parallel algorithm and architecture are described in Section 2 and Section 3, respectively. Finally, Section 4 gives a conclusion.

2. ALGORITHM

The original GEA is described as Equ. (1)-(7).

$$-p \le (m,n) \le p-1 \tag{1}$$

$$0 \le (i,j) \le 2^L - 1 \tag{2}$$

$$CS_{L,j,i} = \sum_{yc=y_a}^{y_b} \sum_{xc=x_a}^{x_b} C(xc,yc) \tag{3}$$

$$0 \le (xc,yc) \le N-1$$

$$y_a = j \cdot N/2^L, y_b = y_a + N/2^L - 1$$

$$x_a = i \cdot N/2^L, x_b = x_a + N/2^L - 1$$

$$SS_{L,j,i,n,m} = \sum_{ys=y_c}^{y_d} \sum_{xs=x_c}^{x_d} S(xs,ys) \tag{4}$$

$$0 \le (xs,ys) \le 2p+N-2$$

$$y_c = n+p+j \cdot N/2^L, y_d = y_c + N/2^L - 1$$

$$x_c = m+p+i \cdot N/2^L, x_d = x_c + N/2^L - 1$$

$$SSAD(m,n) = \sum_{j=0}^{2^L-1} \sum_{i=0}^{2^L-1} |CS_{L,j,i} - SS_{L,j,i,n,m}| \tag{5}$$

$$(m_i,n_i) \in \{(m,n)|SSAD(m,n) \le SSAD_M\} \tag{6}$$

$$MV = \{(u,v)|SAD(u,v) \le SAD(m_i,n_i)\} \tag{7}$$

$$SAD(m,n) = \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} |C(x,y) - S(x+m+p, y+n+p)|$$

The search range is $[-p, p-1]$, (m,n) denotes a search position, and (i,j) is the subblock index. Level is indicated by L, and a block of size $N \times N$ is divided into $2^L \times 2^L$ subblocks of size $(N/2^L) \times (N/2^L)$. The current block data and the search area data are denoted as C and S, respectively. CS is the sum of all pixels within a subblock in current block, and SS is the sum of all pixels within a subblock in a candidate block. Originally, the matching criterion is sum of absolute differences (SAD) for all pixels in the block. Here, we define subsampled-SAD ($SSAD$) as sum of absolute differences between CS and SS. After all the $SSAD(m,n)$ values are calculated, we will find the most probable M motion vectors (m_i,n_i) whose $SSAD$ values are the smallest. The M-th smallest $SSAD$ among all candidate blocks is denoted as $SSAD_M$. Finally, we compute the SAD at the M search positions to find the final motion vector (MV). In our previous work [3], we found that $L = 2$ and $M = 7$ are suitable

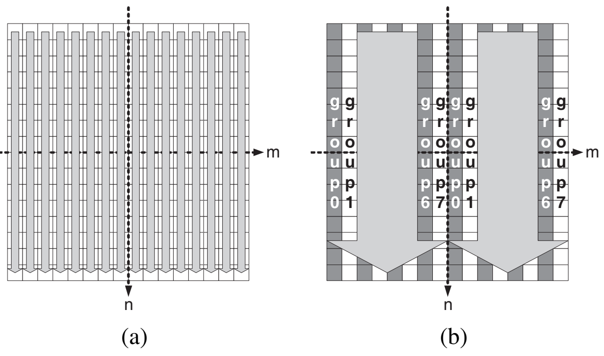

Fig. 1. Scanning order of search positions for $SSAD$ calculation: (a) sequential GEA; (b) parallel GEA with P=8.

parameters for CIF and QCIF under $N = 16$ and $p = 16$ or $p = 32$. We also proposed an architecture with a systolic module and an 16-pel SAD tree to efficiently calculate $SSAD$ and SAD, and with a comparator tree to record the M most probable motion vectors. The comparator tree is designed to match the throughput of generating $SSAD$ values, so the critical path of the comparator tree is roughly proportional to $\log_2^{M+1}$. However, for high-end applications with larger frame size, the search range and the M parameter should be enlarged (e.g. p=64, M=15 or 31) to obtain high video quality. Moreover, our previous architecture computes $SSAD$ sequentially, so the operating frequency must be increased with search range and frame size. Consequently, parallel algorithm and architecture with short critical path are demanded.

In order to compute the $SSAD$'s of several candidates blocks in parallel, we divide them into P groups. Candidate blocks with the same value of $m\%P$ are grouped together, and the the most probable K motion vectors with the smallest $SSAD$ are found separately for each group. Hence, after all the $SSAD$ values are estimated, SAD values of the $K \cdot P$ search positions are further computed to get the final motion vector. Although the $K \cdot P$ most probable candidates do not correspond to the $K \cdot P$ smallest $SSAD$ values within the whole search range, the parallel GEA does not suffer noticeable quality degradation because the $K \cdot P$ globally smallest $SSAD$ values usually belong to different groups. The collection of K candidates in each of the P groups should be similar to the $K \cdot P$ candidates with globally smallest $SSAD$ values. In this way, P duplications of the original GEA architecture can be configured as an array of GEA processing elements (PE's) to support parallel scanning of search positions and parallel calculation of $SSAD$ values. Figure 1 illustrates the scanning order. Besides, K is much smaller than $K \cdot P$, which indicates that the critical path of comparator tree in each GEA PE can be reduced at the algorithmic level.

Many conditions have been tested to verify the quality of our parallel GEA. In our experiments, we embed parallel GEA with P=8 and K=3 into an MPEG-4 simple profile encoder. The resolution of CS and SS is truncated from 12-bit to eight-bit in order to save more area and to reduce the critical path for hardware. The other parameter sets are {CIF 30Hz [-32, +31.5] 384-2048Kbps} and {D1 30Hz H[-64, +63.5] V[-32, +31.5] 1536-8192Kbps}. CIF sequences are Foreman, Hall Monitor, Mobile Calendar, Stefan, and Table Tennis. D1 sequences are two clips from the movie, Crouching Tiger Hidden Dragon. One clip is the scene with two actresses fighting in the courtyard, and the other clip is the leading actor "playing" with the leading actress on bamboos. Compared with FSBMA, the average PSNR losses for the seven sequences are only 0.16, 0.13, 0.05, 0.00, 0.14, -0.02, 0.05dB, respectively. Note that Lagrangian mode decision [4] is applied for both GEA and FSBMA.

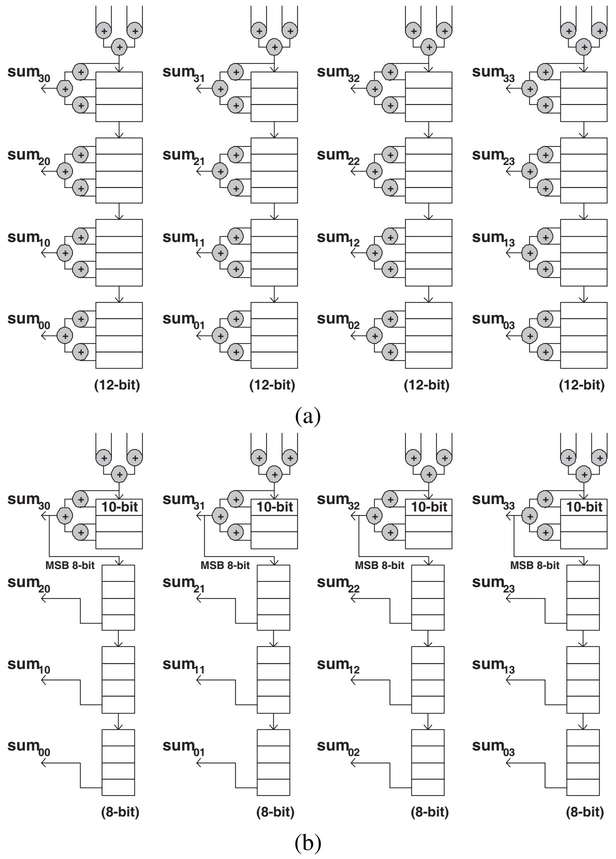

Fig. 2. Systolic module to generate 16 subblock sums of 4×4 pixels: (a) original; (b) proposed.

3. ARCHITECTURE

In this section, N=16, L=2, P=8, K=3, and search range as large as H[-64, +63.5] V[-32, +31.5] are used as an example to explain the parallel GEA design. The specification is D1 30Hz.

3.1. Systolic Module

The purpose of the systolic module is to generate 16 subblock sums of 4×4 pixels in parallel. As shown in Fig. 2, the input is a row of 16×1 pixels. After consecutive 16 rows of pixels are inputted, the 16 subblock sums at search position $(m, -p)$ are produced. The systolic module utilizes vertical data reuse, so the subblock sums at the search positions $(m, -p+1)$-$(m, p-1)$ can be obtained in the following $(2p-1)$ cycles. The improved systolic module not only removes the redundant computation of subblock sums but also reduces the resolution of subblock sums. The gate count of this part is reduced from 6.0K to 4.7K.

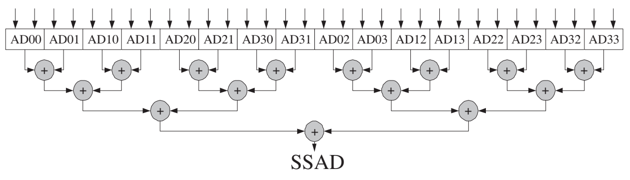

Fig. 3. SAD Tree to compute $SSAD/SAD$

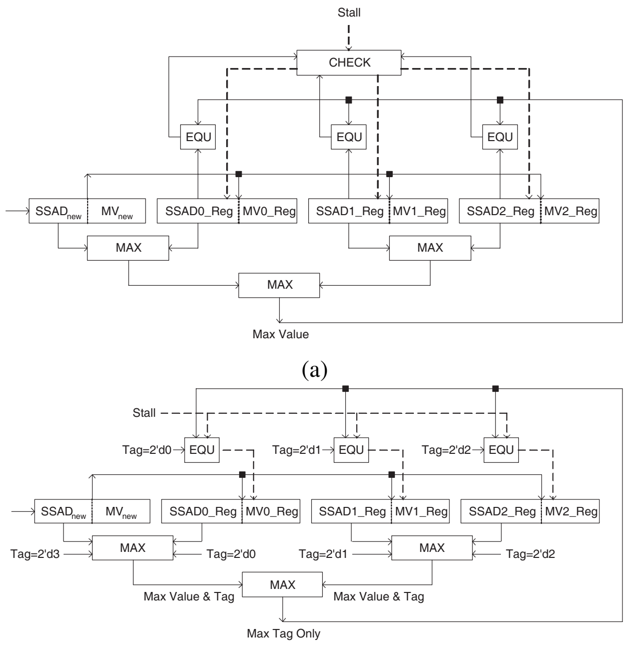

Fig. 4. Comparator tree to find the three smallest $SSAD$ values: (a) original; (b) proposed.

3.2. SAD Tree

The SAD tree is illustrated in Fig. 3, and the goal is to compute $SSAD/SAD$ values. An AD unit computes the absolute difference of two eight-bit samples. When SAD tree is used to generate $SSAD$ values, the inputs are 16 subblock sums of current block and 16 subblock sums of a candidate block. The throughput is the same as the systolic module, i.e. one candidate block per cycle (except the first candidate block at each column of search positions). When SAD tree is used to compute SAD values, the inputs are rows of current block data and search area data, and its output is fetched to a 16-bit accumulator. It takes 16 cycles for one candidate block to compute SAD. Due to the bit-width reduction of CS and SS, the gate count of this part is reduced from 4.6K to 2.8K.

3.3. Comparator Tree

The purpose of the comparator tree is to find the three smallest $SSAD$ values among one group of candidate blocks. The throughput is also matched with the systolic module and the SAD tree. The concept is to keep the up-to-date three smallest $SSAD$ values and their corresponding MV's in the registers, compare the new coming $SSAD$ value with the three stored values, and replace maximum stored values by the new $SSAD$ if it is larger than the new one. Figure 4 illustrates the comparator tree. The MAX unit outputs the larger value of its two inputs, and the EQU unit tells if the two given inputs are the same. The previous architecture shown in Fig. 4(a) finds the maximum $SSAD$ value and feed it back to compare with three stored values to see if a stored value should be replaced. The CHECK unit is to ensure that only one stored value will be replaced if more than one stored values are equal to the maximum. Stall signal should be activated when the invalid $SSAD$ value is generated from SAD tree for the first 15 cycles of a column of search positions. We shorten the critical path in three aspects. First, at the algorithmic level, search positions are divided into eight groups. Originally, if M=24, we will have to find the 24 smallest values, but now only three smallest values in each group are required. Second, the bit-width of $SSAD$ is reduced from 16-bit to 12-bit. Third, as shown in Fig. 4(b), instead of feeding the maximum $SSAD$ value back to compare for replacement, we give each $SSAD$ value an unique 2-bit tag and feed the tag with the maximum $SSAD$ back for comparison. The gate count of this part is reduced from 1.5K to 1.1K.

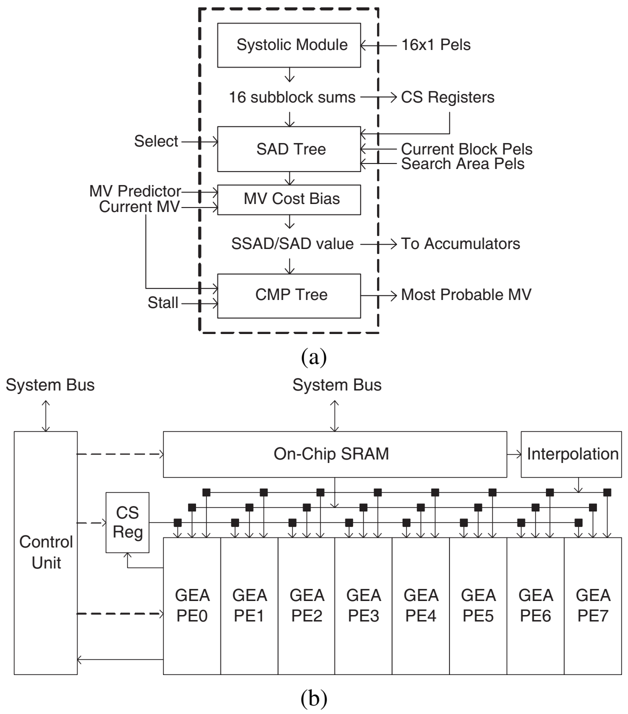

Fig. 5. Illustration of the motion engine: (a) GEA processing element; (b) system block diagram.

3.4. Entire Motion Engine

Figure 5(a) illustrates a GEA processing element (PE). The systolic module, SAD tree, MV cost generator, and comparator tree are configured in cascoding. The MV cost generator, which requires only 0.6K gates, adds a bias of motion information to the distortion function (known as Lagragian method [4]) and provides additional coding gain of 0.2-1.0 dB in PSNR for the MPEG-4 simple profile encoder. The gate count of a GEA PE is 11.3K. Figure 5(b) illustrates the entire ME accelerator. Current block data and search area data are loaded from external SDRAM to on-chip SRAM's. We adopt data reuse of overlapped search area between two horizontally adjacent macroblocks (MB's) to reduce the bus bandwidth from 477 Mbytes/sec to 71 Mbytes/sec. The interpolation circuit is used to generate half-pixels. Besides, thanks to the versatility of SAD tree, advanced prediction (AP) mode (four 8×8-MV's for an MB) is also supported. Inter mode selection between 16×16 and 8×8 configurations of an MB is done after half-pixel ME, and intra/inter mode decision are also included in the accelerator. In general, an MPEG-4 simple profile encoder with our ME accelerator provides better coding performance of 0.5 dB in PSNR than the reference software using FSBMA.

The sequential GEA only utilizes the data reuse in the vertical direction by systolic module to compute the $SSAD$ values. For one column of 64 search positions, 79 rows of 16×1 pixels are fetched, and 1264 bytes of memory access are required. The parallel GEA utilizes not only the vertical but also the horizontal data reuse. As mentioned before, $SSAD$ values of eight columns of search positions are generated in parallel. In order to achieve parallel $SSAD$ calculation, 79 rows of 23×1 pixels (1817 bytes) are fetched. Let us denote the fetched 23×1 pixels from left to right as p0-p22. The first 16 pixels, p0-p15, are sent to PE0, p1-p16 are sent to PE1, ..., and p7-p22 are sent to PE7. In this way, compared with the sequential GEA (1264 bytes for one column), parallel GEA is much more efficient in on-chip SRAM access (1817 bytes for eight columns, i.e. 227 bytes for one column on average). The total bandwidth of on-chip SRAM for $SSAD$ calculation is thus reduced from 6.55 Gbytes/sec to 1.18 Gbytes/sec.

The numbers of cycles to compute CS values, $SSAD$ values, integer SAD values, and half SAD values are 16, 1264, 384, and 58, respectively. Therefore, for processing an MB, about 1730 cycles are required (including pipelines and mode decision). For D1 30Hz, there are 40,500 MB's in a second, so the required frequency is about 70 MHz. We use one 32x128 dual-port SRAM and eight 400x32 dual-port SRAM's to buffer current block data (16×16 pels $\times$ 2) and the search area data (160×80 pels), respectively, for two horizontally adjacent MB's. The advantage of dual-port SRAM's is that the loading of pixels via bus for the right MB of an MB pair and the ME process for the left MB can use different SRAM ports, so that they can be executed at the same time. If only single-port SRAM's are available, the two tasks cannot be operated simultaneously, and the operating frequency should be increased to 100-120 MHz depending on the bus traffic and protocol (assume the bus is 32-bit wide).

3.5. Comparison

We compare our implementation with 1-D semi-systolic FSBMA array architecture [5] due to its high flexibility of search range, scalability of processing elements, and 100% utilization. The results are shown in Table. 1. The gate count of our design is four, eight, and 16 times smaller than 512-PE, 1024-PE, and 2048-PE array, respectively, while the minimum working frequency of our design for D1 30Hz H[-64, +63] V[-32, +31] is 0.42, 0.84, and 1.67 times of the three array configurations. Apparently, our design is more efficient in area and speed. However, the on-chip SRAM access is larger than FSBMA architectures. As for the functionality, our design is the most rich and includes integer ME, half ME, AP mode, and Lagrangian mode decision. The video quality of an MPEG-4 simple profile encoder adopting our ME accelerator is 0.1-0.2 dB worse than that of adopting FSBMA and Lagrangian mode decision, but is significantly better than the reference software.

Table 1. Comparison of ME architectures under the specification of D1 30Hz H[-64, +63] V[-32, +31].

Architecture	8-Parallel GEA	512-PE 1-D Array	1024-PE 1-D Array	2048-PE 1-D Array
Gate Count	113 K	448 K	896 K	1792 K
Working Frequency	70 MHz	166 MHz	83 MHz	42 MHz
SRAM Size	13.312 Kbytes	13.312 Kbytes	13.312 Kbytes	13.312 Kbytes
SRAM Bandwidth	1842 Mbytes/sec	498 Mbytes/sec	249 Mbytes/sec	124 Mbytes/sec
Bus Bandwidth	71 Mbytes/sec	71 Mbytes/sec	71 Mbytes/sec	71 Mbytes/sec
Functionality	Integer ME, Half ME, AP Mode, Lagrangian MB Mode Decision	Integer ME (FSBMA)	Integer ME (FSBMA)	Integer ME (FSBMA)

4. CONCLUSION

This paper presents a new parallel global elimination algorithm and architecture for fast block matching. By rejecting less possible candidate blocks with simplified distortion estimation, only a few most probable candidates are required to determine the final motion vector with fine distortion estimation. The computational complexity of our algorithm is about 10% of the full search. Besides, candidate blocks are divided into independent groups so that the coarse distortion estimation of several search positions can be executed in parallel. A parallel GEA architecture design is also introduced. Many design techniques, such as systolic flow, 2-D data reuse, reuse of overlapped search area, and resource sharing, are proposed to maximize the overall system performance. Our accelerator is much more area-speed efficient than full search architectures and provides better coding performance than the MPEG-4 reference software.

5. REFERENCES

[1] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Transactions on Image Processing*, vol. 4, no. 1, pp. 105–107, January 1995.

[2] X. Q. Gao, C. J. Duanmu, and C. R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 501–504, March 2000.

[3] Y. W. Huang, S. Y. Chien, B. Y. Hsieh, and L. G. Chen, "An efficient and low power architecture design for motion estimation using global elimination algorithm," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002, pp. 3120–3123.

[4] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G. J. Sullivan, "Performance comparison of video coding standards using lagragian coder control," in *Proc. of IEEE International Conference on Image Processing*, 2002.

[5] K. M. Yang, M. T. Sun, and L. Wu, "A family of vlsi designs for the motion compensation block-matching algorithm," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 2, pp. 1317–1325, October 1989.